AIDE, A NEW DIGITAL AUDIO EFFECTS DEVELOPMENT ENVIRONMENT

Victor Lazzarini

Music Technology Laboratory
NUI Maynooth, Ireland
victor.lazzarini@may.ie

Rory Walsh

13 Rue Louis de Cardonnel
Grenoble, France
rorywalsh@ear.ie

ABSTRACT

This paper describes a new rapid development environment for digital audio applications and computer instruments, AIDE (Audio Instrument Development Environment). The system is designed to help users build signal processing applications for use in music, multimedia and sound design. Based on a graphical patching principle, this system generates software using the V and Sound Object libraries. These provide the graphical interface/application framework and sound processing elements, respectively, for stand-alone programmes generated by AIDE. It is envisaged that the system will also generate application components in addition to stand-alone programs. The paper outlines in some detail the elements involved in the software. It discusses how the system is aimed at different types of users with different levels of interaction. The paper concludes with an overview of the typical application development cycle using the system.

1. INTRODUCTION

AIDE is a new software system for computer instrument and stand alone audio software development in C++. It is intended for use by musicians, composers and programmers, as it provides different levels of user interaction. For those with no prior experience of programming, the software can be used as a pedagogical system, in which users can learn the intricacies of programming. Application development can be done through a user-friendly graphical approach. The system also provides lower-level interaction for more experienced programmers.

The first version of the system provides support for the development of stand-alone applications, but it is hoped that in the future users will also be able to develop software components. These would comprise Pure Data(PD) [1], MaxMSP [2] classes, as well as VST, LADSPA and DirectX plugins. AIDE is designed to assist all aspects of the development of practical tools for music signal processing.

2. SYSTEM FEATURES

The most important feature of AIDE is its versatility. As a basic tool it provides users with a way to create their own standalone audio applications. It is foreseen that users will employ the system to create task specific applications, such as computer instruments for works of electroacoustic music, research tools for audio processing, and educational tools for teaching computer music. The benefit of this is that the applications will be small in size and very easy to set up as the generated executable files do not need any third party libraries to run. The 'ease of setup' factor is very useful to musicians performing computer music. No more will they have to preinstall complete software to run their patches, the only software they will have to install is the executable which runs the process.

Work is now underway to allow for different types of target applications, not merely standalone executables. The generation of PD/MaxMSP externals is possible through the employment of Flext [3], a C++ layer for Max/MSP and PD externals. However, for now external objects are limited to a very basic interface. LADSPA, VST and DirectX plugins are other varieties of output that are planned for the future. This would provide users with a means to creating their own custom plugins, thus stretching the realms of user interaction found in many of today's audio processing systems.

While the above examples illustrates how AIDE proves a very useful tool in the development of audio processing software, it must not be overlooked that it also provides users with access to low level signal processing operations. This type of access is often not available with other audio systems, in particular commercially available systems. While these systems may provide users with pre-built effects such as reverbs, delays, and filters, they do not allow the user to take these effects apart and rebuild them to their own specifications.

3. TECHNICAL ASPECTS

AIDE is being developed for Windows using the Borland© C++ Builder. It is hoped that, with the release of Borland's Kylix© [4] a new Linux C++ development environment, a version for Linux will also be made available. The system works by creating the appropriate C++ code whenever the user places a new SndObj object or graphical component in the main patcher window. AIDE makes use of the freeware multi platform GNU gcc compiler to compile the standalone executables, therefore even though an application is created in Windows, the multi-platform makefile may be run on any Linux PC or Mac OSX providing that the correct libraries are installed.

3.1. The V library

The V GUI Library [5] is used to create the GUI interface for standalone applications. V is a C++ Graphical User Interface Framework designed to provide an easy to use system for building GUI applications. The framework is small, and provides all the tools one would need for building intuitive graphical user interfaces. V has also been designed to be portable and, currently, versions for Linux, Microsoft Windows, and OS/2 exist. The

V framework is freely available for use by anyone under the terms of the GNU Library General Public License.

3.2. The SndObj library

AIDE uses the Sound Object Library [6] to provide the audio processing operations for the generated software. The generated code is fully portable across Windows, Linux/Unix (with OSS), Irix and MacOS X. The SndObj library provides around 100 classes that can be used for time- and frequency-domain signal processing, as well as sound and MIDI input/output. With its `SndThread` class, it can also manage audio processing threads, something particularly important for this software.

The library is normally used by this software as a toolkit, but also serves as a framework for class development. The software, with its code generation capabilities, includes the possibility of user-defined custom SndObj classes. It is expected that AIDE will be used in the further development of the library.

4. LEVELS OF INTERACTION

The system was designed to be used at three different levels of interaction:

- **Introductory**: The first level is directed at the novice user, who has no prior knowledge of audio processing systems yet wants to learn about the many ways to manipulate and transform sounds digitally. By using AIDE they can start learning from the very beginning. By following interactive tutorials in which they follow flow charts of classic processing techniques they will begin see the different ways in which the classic techniques can be realised using the system. They may then start to create their own applications by recreating these flow charts on screen in a modular based fashion to create new applications in which these techniques are embedded.
- **Advanced:** the second level is directed at users with prior knowledge of processing techniques. They may use AIDE to create and develop new techniques. As the system comes with a complete range of audio processing tools in the form of SndObj classes, it should make it possible to create or recreate a huge array of processing techniques. The user may then embed these new techniques into standalone software or plugins.
- **Developer:** The third level of interaction aims to support users experienced in processing techniques and in C++ programming. For them, the software will help them to realise more complex audio applications and enable fast development with code re-use. Skeleton applications which include just the GUI elements such as Menu items, buttons, scrollbars, etc, can be augmented by the user's own C++ code. This can be edited from within the application in a code text window. After the code is modified, it can be compiled as normal from within the Application Builder or from the command line. In this way users can benefit from code reuse, esp. when developing a graphical user interface for their application. This particular level of interaction provides experienced users with a great tool for researching new custom built audio processing techniques.

5. PROGRAM LAYOUT

AIDE incorporates a modular design 'flow chart' system, similar to many current audio processing systems, such as PD/MaxMSP, OSW [7] and CPS [8]. This system was chosen above others because of its clarity and simplicity of use. In addition, it is hoped that users of Pd/MaxMSP will not have many problems in adapting to AIDE as it incorporates the same 'patcher' paradigm. So as in many other audio processing systems the user simply drags and drops classes into the main patcher window to instantiate objects. These are then connected together in typical patch chord fashion. When the user starts a session with the AIDE they are presented with three main windows:

1) The Main Patcher Window: this contains the graphical representation of the audio processing flow.
2) The Source Code Editor: this contains the entire source for the project.
3) The GUI Layout, Data Structure and compiler output window: this contains the GUI designer where the user implements the graphical user interface for their application. It also contains the processing order of the classes on screen and also informs the users of which patch chord goes where. Finally it contains the compiler output to inform users of whether or not their projects compiled correctly.

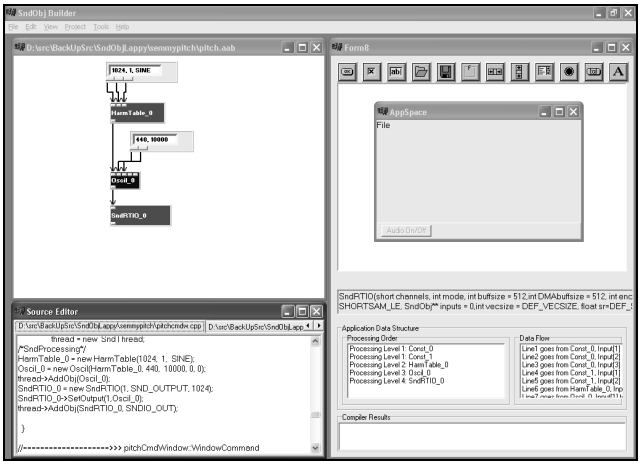

Figure 1: *Main Program Window*

6. IMPLEMENTING A SIMPLE APPLICATION

To illustrate a typical session it is best to implement a simple application, such as a Schroeder Reverb Unit. The user, upon opening the main program interface, begins by simply inserting the needed sound objects into the 'patcher' window. These objects are connected together with patch chords to determine the data flow of the application.

Figure 2 shows an implementation of our simple 'Schroeder reverb' unit. As you can see from the diagram the flow chart is quite simple to follow. There are four parallel comb filters, each with a constant for the gain and delay time parameters connected to two cascading allpass filters. The allpass filters also have two parameters, again for the gain control and the delay time. The SndRTIO objects handle real time audio IO and the SndIn object

captures the audio stream and makes it accessible to other SndObjs. Each of the sound objects are named in the same way as they appear in the SndObj library so as not to confuse a user who wishes to make the move from using the AIDE to using the SndObj Library in another programming environment outside of AIDE.

After the signal processing patch is created, a Graphical User Interface for the application can be generated using the GUI designer. Again by a simple process of drag and drop a nice user friendly interface can be created for the application. In the above patch, constants have been used to provide parameters for the different class members. However if the 4 comb filter gains were made variable, they could be linked to GUI components, such as a scroll bar, as in Figure 2.

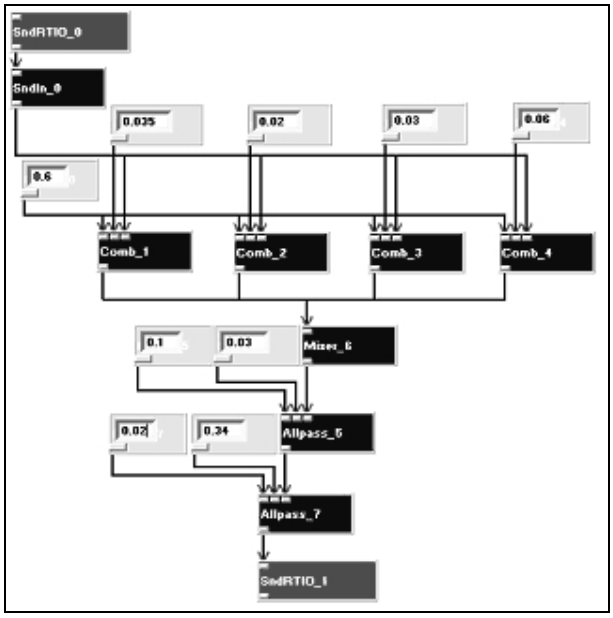

Figure 2: *Schroeder Reverb Implementation*

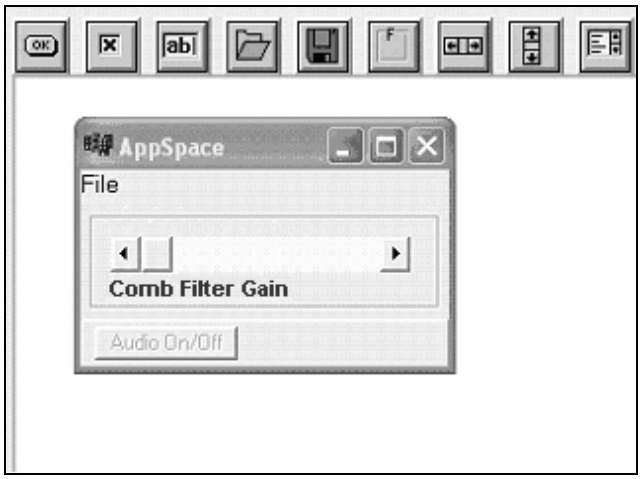

Figure 3: *A Simple GUI Interface*

Each time a user places, deletes, or moves a graphical element, be it a sound object or a GUI object, C++ code is generated by AIDE to correspond with each new object. This C++ code is visible to the user through the source code editor giving them the opportunity to take a glimpse at how the code is structured and placed together. The automatically generated source code is put together in the most user friendly fashion possible, to give the user a clear idea of how the application works.

After completing the graphic interface, the user can proceed to the compilation of the new software. When the user compiles the project AIDE runs the project *makefile*. If there are errors in the compilation of the application, the user may view them through the compiler output window which is integrated into the software main interface. Providing that the source code has no user errors AIDE will compile everything into a standalone executable. As stated earlier the source code for each project and makefiles are cross-platform and can be compiled on many different platforms.

7. AIDE IN THE CONTEXT OF CURRENT AUDIO PROCESSING SYSTEMS

While AIDE obviously contains a certain likeness, in terms of user interaction, with other software systems such as PD, Max/MSP and OSW, the similarities stop there. There is no audio engine in AIDE, thus there is no run-time environment for the execution of the newly created programs. Instead, each patch must be carefully planned and implemented by the user before compilation in order for it to run and compile correctly. Through this approach users will find it easier to develop highly structured applications. None of the above systems are designed to allow users access to the actual processing code. In some ways this is where AIDE is most innovative. By offering access to the C++ code, users are being encouraged to look at the algorithms and classes for each of the SndObjs employed in their patch. With the freedom AIDE offers for exploration of processing classes, it is not difficult to see how the system can become a useful tool in the development and research of audio instrument design.

Another original aspect of this system is that it uses a platform- independent GUI C++ library for application development. While other software does offer users the options of creating GUI interfaces to control processing parameters, these are in general intricately connected to the main application, which sometimes is not desirable. Using a C++ application framework is possibly a more flexible and open way to provide GUI support for the generated software.

Among the comparable systems, CPS, a new realtime processing environment, appears to share some of the aims of AIDE. In this system, users can translate their graphical patches into C++ or Java source code. This source code can then be used together with a supplied SDK to create standalone applications. Unlike AIDE, however, CPS is not a development environment as such, lacking the compiler, text-editing and application-building support. CPS seems to be primarily a synthesis and processing system, with some secondary application development features. In addition, as opposed to AIDE, it is not free software.

Finally, it is also important to point out that, while all other systems discussed here provide their own basic signal processing and GUI functionality, AIDE depends on external libraries. This model allows for a more flexible evolution of the whole system. Separate upgrades of the libraries will add new functionality to AIDE, without the need for new versions of the program itself.

8. CONCLUSION

AIDE has been used very successfully by the authors and others in beta-testing. It has been employed in the development of computer instruments for live electroacoustic music, for sound processing tutorial materials and in general music applications. Further work will possibly involve also the addition of different types of output (plugins/components), as well as preview capabilities, whereby the sound processing operations can be tested prior to compilation. This would involve the design of a light-weight processing engine, based on the processing thread management services provided by the SndObj library. It is envisaged that, with the addition of these features, the system will become a comprehensive tool for audio application development. The beta-versions of the application, including source code and examples will soon be available on-line at the NUIM Music Technology Laboratory site:
`http://www.may.ie/academic/music/musictec.`

9. ACKNOWLEDGEMENTS

The research project that led to the development of AIDE was carried out in the La Villa Media Institute, Grenoble, France, under the auspices of the French Ministry of Education. The authors would like to thank the Institute, its staff, and its Director, Bernard Cornu, for their support and encouragement.

10. REFERENCES

[1] Puckette, M., "Pure Data," in *Proc. International Computer Music Conference*, San Francisco, pp. 269–272, 1996.

[2] Puckette, M., "Max at Seventeen," *Computer Music Journal* 26 (4), MIT Press, Cambridge, Mass., pp. 31–43, 2002.

[3] Grill, Thomas. "Flext, C++ layer for MaxMSP and pd externals." `http://www.kapazitaeten.net/Pd/ext/flext`

[4] Borland Software Corporation. "Kilyx: Rapid e-business Development for Linux." `http://www.borland.com/kylix/index.html`

[5] Wampler, B. "V - A Freeware Portable C++ GUI Framework for Windows, X, and OS/2." `http://www.objectCenTral.com`

[6] Lazzarini, V, "The Sound Object Library," *Organised Sound* 5 (1), Cambridge: Cambridge Univ. Press., pp. 35–49, 2000.

[7] A. Chaudhary, A. Freed, and M. Wright, "An Open Architecture for Real-Time Music Software," *Proceedings of International Computer Music Conference*, Berlin, Germany, 2000.

[8] Douglas Keislar (Ed.). "Products of Interest," *Computer Music Journal* 28 (3), Cambridge Mass, 2003, pp. 114–115.